In the Specification:

Page 1 Under the title, please insert the following:

-- **CROSS REFERENCE TO RELATED APPLICATIONS**

This application claims the priority of PCT/GB2005/001300 filed on April 1, 2005, and, GB 0407544.6 filed on April 2, 2004, the entire contents of which are hereby incorporated in total by reference. --

Page 8, line 4 Please add the following:

Description of the Drawings

An embodiment of the present invention will now be described, by way of further example only, with reference to the accompanying drawings in which: Figure 1 illustrates examples of monolithic and micro kernel architectures;

Figure 2 illustrates a procedure for waiting on a fast mutex in accordance with an embodiment of the present invention;

Figure 3 illustrates a procedure for releasing a fast mutex in accordance with an embodiment of the present invention; and

Figure 4 illustrates a procedure for optimising both waiting for and releasing a fast mutex in a single processor (non-SMP) system.

Page 13, line 19 Please add the following:

This procedure is shown in Figure 2.

Page 13, line 23 Please add the following:

In the case where there is no contention, this simply reduces to two variable assignments. On non-SMP (symmetrical processing) systems this can be and has been further optimised by simply disabling interrupts rather than locking the kernel while checking iHoldingThread. This procedure is shown in Figure 4.

Page 15, line 14 Please add the following:

This procedure is shown in Figure 3.

Page 15, line 15 Please add the following:

In the case where there is no contention, this simply reduces to two variable assignments. Again, on non-SMP systems this can be and has been further optimised by disabling interrupts rather than locking the kernel while checking the iWaiting flag. This optimization is shown in Figure 4. The iWaiting flag will have been set if another thread had attempted to acquire the mutex while the current thread held it. It will also have been set if the thread's timeslice has expired (the

round-robin with other equal priority threads is deferred until the fast mutex is released) or if an attempt was made to suspend or kill the thread. The latter scenario is handled in a similar way to the case where the thread was executing in a thread critical section (ie iCsCount nonzero). The deferred operation is processed when the fast mutex is released, hence the check of iCsCount and iCsFunction in the mutex release code.